Current and Emerging Topics in Software Architecture (ECSA 2010 Workshops Summary)

Danny Weyns
Katholieke Universiteit
Leuven, Belgium
danny.weyns@gmail.com

Rafael Capilla
Universidad Rey Juan Carlos
Madrid, Spain
rafael.capilla@urjc.es

SUMMARY

Since 2004 in St. Andrews (Scotland, U.K.), ECSA the European Conference on Software Architecture (formerly EWSA, the European Workshop on Software Architecture) has been considered as an important meeting point for researchers and practitioners on the topic of software architecture. ECSA has matured from a workshop format to a full software engineering conference in the subfield of software architecture.

This year, ECSA has become more ambitious and expanded its scope and schedule up to four full days. The program includes a series of tutorials, a doctoral mentoring program, and four full-day workshops. New and existing software challenges have led to a variety of trends in software architecture research, which makes the conference and workshops more attractive and promotes the discussion on current and emerging topics.

Based on the scientific and technical interest of the topics, the innovativeness of workshop topics, and the capacity of the conference workshop program, the workshop co-chairs selected four workshops from the nine submitted proposals. We summarize the aims and goals of each workshop and the contributions accepted for the four workshops:

- 2nd International Workshop on Software Ecosystems (EcoSys). *Piers Campbell, Faheem Ahmed, Jan Bosch, Sliger Jansen.*
- 1st International Workshop on Measurability of Security in Software Architectures (MeSSa). *Reijo Savola, Teemu Kranstén, Antti Evesti*.
- 8th Nordic Workshop on Model Driven Software Engineering (NW-MODE). *Andrzej Wąsowski, Dragos Truscan, Ludwik Kuzniarz*.
- 1st International Workshop on Variability in Software Product Line Architectures (VARI-ARCH). *Alexander Helleboogh, Paris Avgeriou, Nelis Boucke, Patryck Heymans.*

The ECSA 2010 Workshop co-chairs would like to thanks all workshop organizers for their effort and enthusiasm to attract submission in different software architecture research topics and make the ECSA 2010 workshops a success.

ECSA 2010, August 23-26, 2010, Copenhaguen, Denmark.

1. SOFTWARE ECOSYSTEMS (EcoSys)

Software ecosystems are a novel area which encourages external developers to use an organizations' software platform and contributes in product development. This establishes a community that further accelerates the sharing of knowledge, content, issues, expertise, and skills. There has been a significant shift in the development strategies for software product development in the past decade. The traditional single software product development strategy has been replaced by multiple product development, which employs a common architecture using the concept of software product lines. However, the past few years have seen a further shift as the software product line concept moves towards increased openness through direct involvement with external developers outside the domain of organizations. This paradigm shift has been termed as the development of software ecosystems. During the workshop, 12 papers are presented that broadly cover the various aspects of software ecosystems, such as architecture, social networking, modeling, business considerations for software ecosystems, mobile and virtual software and managing software ecosystem from organizational perspectives.

Cataldo and Herbsleb [1] present the concept of interface translucence as an architectural mechanism that seeks to overcome challenges faced by transparency and modular system design. Hunink et al. [2] elaborate a method for creating complete and encompassing domain specific taxonomies; it offers a structured manner in order to engineer an industrial taxonomy within a specific domain. Dhungana et al. [3] compare software ecosystems and natural ecosystems to present an agenda for further research by analyzing some key characteristics of both types of ecosystems. They further discuss the regulatory factors and mechanisms existing in nature, and then deduce key challenges that need to be dealt with, in order to achieve healthy operation of software ecosystems. Bosch [4] discusses some key architectural challenges which UI stability, integration of user interfaces, workflow and data, security and reliability, incorporating new functionality and, finally, refactoring the platform to replace proprietary functionality with commercial or open-source software components. Anvaari and Jansen [5] compare the openness and architectures of five mobile platforms to provide further insight into successful openness strategies for software ecosystem growth. The overall aim of this research was identification of openness strategy in mobile platforms based on the software architecture of the platforms. van den Berk et al. [6] present a model that describes the key characteristics of a Software Ecosystem and can be used as a tool for strategic guidance in software ecosystem. Pettersson et al. [7] identify the need for precise process modeling. Their work elaborates on experiences gained from the analysis of a software ecosystem for mobile learning and brings up several aspects and insights for this

particular domain. They also propose an initial reference model for the mobile learning domain and an outline for an analysis method for domain specific software ecosystems. dos Santos and Werner [8] revise the concept of components in software engineering through a socio-technical construction based on the discussion involving technical literature authors' points of view, experts' opinions, and the perspectives of the authors of this work. Seichter et al. [9] propose a new approach for handling the diverse software artefacts in ecosystems by adapting features from social network sites. Capuruço and Capretz [10] present a recommender model that was developed incorporating time-dependent and social-aware recommendations, an interaction-based social network quantifier to identify the proximity of their members, and a modified search algorithm to optimally reach members. Campbell and Ahmed [11] present a three dimensional view of the software ecosystem model examine the role played by each of the three central pillars; business; architecture; and social aspects. They further highlight their relationships and conclude that this study will help in further aiding understanding of the overall engineering process of ecosystem software. McGregor [12] presents an analysis technique that uses the economic notion of a transaction to examine the transfers between entities in Software Product Lines. The results of this analysis are data that can be used to evaluate and structure a given organization. An example case study is also included to support the findings of the paper.

2. MEASURABILITY OF SECURITY IN SOFTWARE ARCHITECTURES (MeSSA)

The growing complexity of service-centric systems has increased the need for pertinent and reliable software security and trusted system solutions. Systematic approaches to measuring security in software architectures are needed in order to obtain sufficient and credible proactive evidence of the security level or performance of a system, service or product. The systematic definition of security metrics and security assurance metrics is a young field that still lacks widely accepted definitions of metrics and applicable measuring techniques for design-time and run-time security monitoring.

In [13], Antonino et al. introduce a method called SiSOA for security evaluation of existing complex service-oriented systems at architectural level. The method is based on reverse engineering techniques and utilizes a knowledge base. Blasi et al. [14] describe and discuss experiences from deployment of security metrics-driven adaptive security solutions of a distributed middleware. In [15], Evesti and Pantsar-Syväniemi introduce a micro-architecture for security adaptation and associated context information taxonomy for smart spaces. The micro-architecture contains six execution phases, one of which being context monitoring. Groven et al. [16] present two quality assessment models, OpenBRR and QualOSS, which are compared in the context of a telephone private branch exchange case study. In [17], Halonen and Hätönen discuss security management in the context of complex communication systems, and the authors propose coherent measurement of various technical aspects of security and utilization of security impact metrics. Krautsevich et al. [18] discuss a basic model for formal description and analysis of security metrics, where dependencies of metrics and attacker models are investigated. Mellado et al. [19] compare widely-known security design approaches for software products using metrics. In [20], Nguyen-Tong et al. present an artificial diversity security model for metamorphosis of attack surface called Metamorphic Shield and applied to an incremental attack against instruction set randomization. In [21], Sääskilahti and Särelä outline a risk identification method based on Value Chain Dynamics Toolkit and applied to risk analysis of Host Identity Protocol. The method offers benefits in knowledge transfer, structuring of interviews and visualization of value chains. In [22] Sullivan et al. propose trust-terms ontology for various components and concepts that comprise ICT security and trust. The ontology helps in gaining a better understanding of trust and security requirements and in identifying more precise measurability criteria. Suomalainen et al. [23] describe a novel security architecture for smart spaces enabling heterogeneous devices to share data in controlled manner. Centralized information brokering device is used to measure security level of published information. Finally, in the invited paper [24], Kanstrén describes a taxonomy-based approach for creation of an abstraction layer for available measurements from the requirements, and applied to a security assurance case of Push E-mail.

3. MODEL-DRIVEN SOFTWARE ENGINEERING (NW-MODE)

Model driven software development approaches (MDE,MDA, MDD) have matured and grown out from an academic research community to gain wider industrial adoption. They are now perceived as one of the mainstream technologies to improve the productivity of software teams and the quality of products. Model-driven engineering can bring important benefits to software architectures; however there are still challenges to be addressed in order to realize a successful integration. Therefore, the relationship between model-driven paradigm and the software architecture field require further study.

In [25], Hamid and Krichen address the modeling of reconfigurable embedded systems by firstly, building modeling tools to help specify and configure reconfigurability issues; and secondly by integrating a Model-Based Development approach. Kou et al. [26] present a metamodel called SoaML4Security, which introduces QoS concepts into Service oriented Modelling Language (SoaML) in order to support the modelling of security aspect which can support Model Driven Engineering (MDE) for service-oriented applications. In [27],Mu et al. describe the use of several metamodels for complex specification languages. In [28], Peltonen et al. enhance a modeling tool with model manipulation to facilitate the actual modeling work. Rauf et al. [29] address the differences between Representational-State Transfer (REST) and Remote Procedure Call (RPC) web services in the context of web service compositions and motivates the need for new designing techniques that lead to RESTful interfaces. In [30], Mellegard and Staron report results from a case-study of the development of embedded software at a Swedish vehicle manufacturer. Störrle [31] shows an integrated approach to incorporate the whole UI development life cycle, and support a wide range of levels of granularity and abstraction. It allows closer collaboration between different user groups like graphic designers and software developers by integrating traditional pen-and-paper based methods with contemporary MDA-based CASE tools. Finally, in

[32] Störrle explores the problems and possibilities associated with detecting clones in UML domain models.

4. VARIABILITY IN SOFTWARE PRODUCT LINE ARCHITECTURES (VARI-ARCH)

Critical challenges remain in the crossing of software product lines and software architectures, in particular with respect to variability in software product line architectures. One example challenge in this context is to determine suitable "variability viewpoints" for the specific stakeholders of a product-line architecture. The fact that different stakeholders typically have different concerns w.r.t. variability (e.g. typically only a part of the total variability concerns a particular stakeholder), presents a specific challenge in this context that receives insufficient attention nowadays. Another challenge is that existing work on variability in product line architectures is typically focused on component-and-connector models. Variability in other architectural models such as deployment models, information models, or development models is currently under-investigated. Also the relation between variability and other qualities such as performance and scalability is a challenge that requires further investigation.

Although primarily studied in the context of product lines, variability is a key fact of most systems and therefore of their architectures. Thus it is essential for the Architect to have suitable tools for representing, managing and reasoning about variation. Hilliard [33] presents a simplified model of variation and then explores the consequences of that model for the representation of variation as a part of architecture description, using the conceptual foundation of ISO/IEC 42010 (the revision of IEEE 1471:2000). In [34], Simidchieva and Osterweil describe an approach that considers variation in systems and system architectures according to the kind of relation among the variants in the software family. The approach highlights why it is beneficial to consider such different variation relations separately and gives examples of what these relations may be. Abbas et al. [35] propose a novel variability mechanism that self-optimizes product-line instances. Also, in [36] Galvao et al. present a model for the specification of variability design rationale and its application to the modelling of architectural variability in software product lines. In [37], Geertsema and Jansen present a Software Product Line Infrastructure (SPLI) to increase reuse of software efforts by widening the software product line scope and supporting the reuse of application design via step-wise refinements. The SPLI takes a bottom-up approach by structuring product features in highly reusable software components called Active Components. A model-driven engineering approach is used in which application design is specified using domain-specific models and variability models. Variability within Active Components is bound during product derivation by executing model-to-artifact transformations. Finally, Galster [38] focuses on investigate what types of variability exist in service-oriented software architectures and suggest a way for representing variability and a formalization mechanism.

REFERENCES

All references are published in the proceedings of workshops organized in conjunction with 4th European Conference on Software Architecture (ECSA2010), Copenhagen. In particular, [1] to [12] are published in the proceedings of the 2nd International Workshop on Software Ecosystems; [13] to [24] are published in the proceedings of the 1st International Workshop on Measurability of Security in Software Architectures; [25] to [32] are published in the proceedings of the 8th Nordic Workshop on Model Driven Software Engineering; and [33] to [38] are published in the proceedings of the 1st International Workshop on Variability in Software Product Line Architectures.

[1] Cataldo, M., Herbsleb, J.D., "*Architecting in Software Ecosystems: Interface Translucence as an Enabler for Scalable Collaboration*".

[2] Hunink, I., van Erk, R., Jansen, S., Brinkkemper, S., "*Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens*".

[3] Dhungana, D., Groher, I., Schludermann, E., Biffl, S. "*Software Ecosystems vs. Natural Ecosystems: Learning from the Ingenious Mind of Nature*".

[4] Bosch, J., "*Architecture Challenges for Software Ecosystems*".

[5] Anvaari, M., Jansen, S., "*Evaluating Architectural Openness in Mobile Software Platforms*".

[6] van den Berk, I., Jansen, S., Luinenburg, L., "*Software Ecosystems: A Software Ecosystem Strategy Assessment Model*".

[7] Pettersson, O., Svensson, M., Gil, D., Andersson, J., Milrad, M., "*On the Role of Software Process Modeling in Software Ecosystem Design*".

[8] dos Santos, R.P., Werner, C.M.L., *"Revisiting the Concept of Components in Software Engineering from a Software Ecosystem Perspective"*.

[9] Seichter, D., Dhungana, D., Pleuss, A., Hauptmann, B., "*Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens*".

[10] Capuruco, R.A.C., Capretz, L.F., *"Integrating Recommender Information in Social Ecosystems Decisions*".

[11] Campbell, P.R.J., Ahmed, F., "*A Three- Dimensional View of Software Ecosystems*".

[12] McGregor, J.D., "*A Method for Analyzing Software Product Line Ecosystems*".

[13] Antonino, P., Duszynski, S., Jung C., Rudolph M. "*Indicator-based Architecture-level Security Evaluation in a Service-oriented Environment***".**

[14] Blasi, L., Savola, R. M., Abie H., Rotondi D. "*Applicability of Security Metrics for Adaptive Security Management in a Universal Banking Hub System*".

[15] Evesti A., Pantsar-Syväniemi S. "*Towards Micro Architecture for Security Adaptation*".

[16] Groven, A. K., Haaland, K., Glott R., Tannenberg A. "*Security Measurements within the Framework of Quality Assessment Models for Free/Libre Open Source Software*".

[17] Halonen P., Hätönen K. "*Towards Holistic Security Management through Coherent Measuring*".

[18] Krautsevich, L., Martinelli F., Yautsiukhin A. "*Formal Approach to Security Metrics -- What does "More Secure" mean for you?*".

[19] Mellado, D., Fernández-Medina E., Piattini M. "*Comparison of Software Design Security Metrics*".

[20] Nguyen-Tong, A., Wang, A., Hiser, J. D.,. Knight, J. C., Davidson J. W. "*On the Effectiveness of the Metamorphic Shield*".

[21] Sääskilahti J., Särelä M. "*Risk Analysis of Host Identity Protocol -- Using Risk Identification Method Based on Value Chain Dynamics Toolkit*".

[22] Sullivan, K., Clarke J., Mulcahy B. P. "*Trust-terms Ontology for Defining Security Requirements and Metrics*".

[23] Suomalainen, J., Hyttinen P., Tarvainen P. "*Secure Information Sharing between Heterogeneous Embedded Devices*".

[24] Kanstrén, T. et al. "*Towards a Taxonomy-based Abstraction Layer for Dynamic Security Assurance Measurements*".

[25] Hamid B., Krichen F. "*Model-Based Engineering for Dynamic Recon_guration in DRTES*".

[26] Kou S., Babar M. A., Sangroya A., "*Modeling Security for Service Oriented Applications*".

[27] Mu L., Gjosaeter T, Prinz A., Skjelten Tveit M. "*Specification of Modelling Languages in a Flexible Meta-model Architecture*".

[28] Peltonen J., Felin M., Vartiala M. "*From a Freeform Graphics Tool to a Repository Based Modeling Tool*".

[29] Rauf I., Ruokonen A., Systa T., Porres I. "*Modeling a Composite RESTful Web Service with UML*".

[30] Mellegard N., Staron M. "*Characterizing Model Usage in Embedded Software Engineering: A Case Study*".

[31] Störrle H. "*Model Driven Development of User Interface Prototypes: An Integrated Approach*".

[32] Störrle H. "*Towards Clone Detection in UML Domain Models*".

[33] Hilliard, R., "*On representing variation*".

[34] Simidchieva, B., Osterweil, L., "*Categorizing and Modeling Variation in Families of Systems: a Position Paper*"

[35] Abbas, N., Andersson, J., Löwe, W., "*Autonomic Software Product Lines*".

[36] Galvao, I., van den Broek, P., Aksit, M., "*A Model for Variability Design Rationale in SPL*".

[37] Geertsema, B., Jansen, S., "*Increasing Software Product Reusability and Variability using Active Components: a Software Product Line Infrastructure*".

[38] Galster, M., "*Describing Variability in Service-oriented Software Product Lines*".